SHAvite-3 — A New and Secure Hash Function Proposal

Orr Dunkelman

Département d'Informatique
École Normale Supérieure

France Telecom Chaire

12th of January 2009

Joint work with Eli Biham.

Outline

Outline

SHAvite-3 (SHAvite-Shalosh)

- A SHA-3 candidate designed to be secure and efficient.
- Uses two compression functions: C_{256} and C_{512}.
- The compression functions are iterated in HAIFA to become a hash function.
- Using HAIFA, supports salts (nonces/randomized hashing), variable digest length, while maintaining security.
- The compression functions are designed using known and understood components: Feistel structure, AES-round, and LFSRs.

HAsh Iterative FrAmework (HAIFA)

- Major features:
  - Supports salts (defines families of hash functions).
  - Supports variable output size.
  - Offers as good security properties as can be.
  - Strong backward compatibility.
  - All suggested modes can be realized as HAIFA.

The HAIFA Construction

- Accepts as inputs:
  - A chaining value (of size m_c)
  - A message block (of size n)
  - A bit counter (of size b)
  - A salt (of size s)

$$C : \{0,1\}^{m_c} \times \{0,1\}^{n} \times \{0,1\}^{b} \times \{0,1\}^{s} \rightarrow \{0,1\}^{m_c}.$$

The HAIFA Initialization

- Let m be the target digest size.
- Let IV be a general initial value.
- $IV_m = C(IV, m, 0, 0)$.

The HAIFA Computation

- Take M, the message, and pad it:
  - Pad a single bit of 1.
  - Pad as many 0 bits as needed such that the length of the padded message (with the 1 bit and the 0's) is congruent modulo n to $(n - (t + r))$.
  - Pad the message length encoded in t bits.
  - **Pad the digest size encoded in r bits.**
- Set $h_0 = IV_m$
- For $i = 1, 2, \ldots, l$ compute $h_i = C(h_{i-1}, M_i, \#bits, salt)$. When the processed block contains only padding, use $\#bits = 0$.
- Truncate h_l to m bits.

SHAvite-3_{256}

- Uses the C_{256} compression function,

SHAvite-3_{256}

- Uses the C_{256} compression function,
- which is a Davies-Meyer transformation of the block cipher E^{256},

SHAvite-3_{256}

- Uses the C_{256} compression function,
- which is a Davies-Meyer transformation of the block cipher E^{256},
- which is a Feistel block cipher, with 12 rounds,

SHAvite-3_{256}

- Uses the C_{256} compression function,
- which is a Davies-Meyer transformation of the block cipher E^{256},
- which is a Feistel block cipher, with 12 rounds,
- where each round is composed of three AES rounds.

SHAvite-3_{256}

- Uses the C_{256} compression function,
- which is a Davies-Meyer transformation of the block cipher E^{256},
- which is a Feistel block cipher, with 12 rounds,
- where each round is composed of three AES rounds.
- The message expansion combines both AES rounds and LFSR behavior.

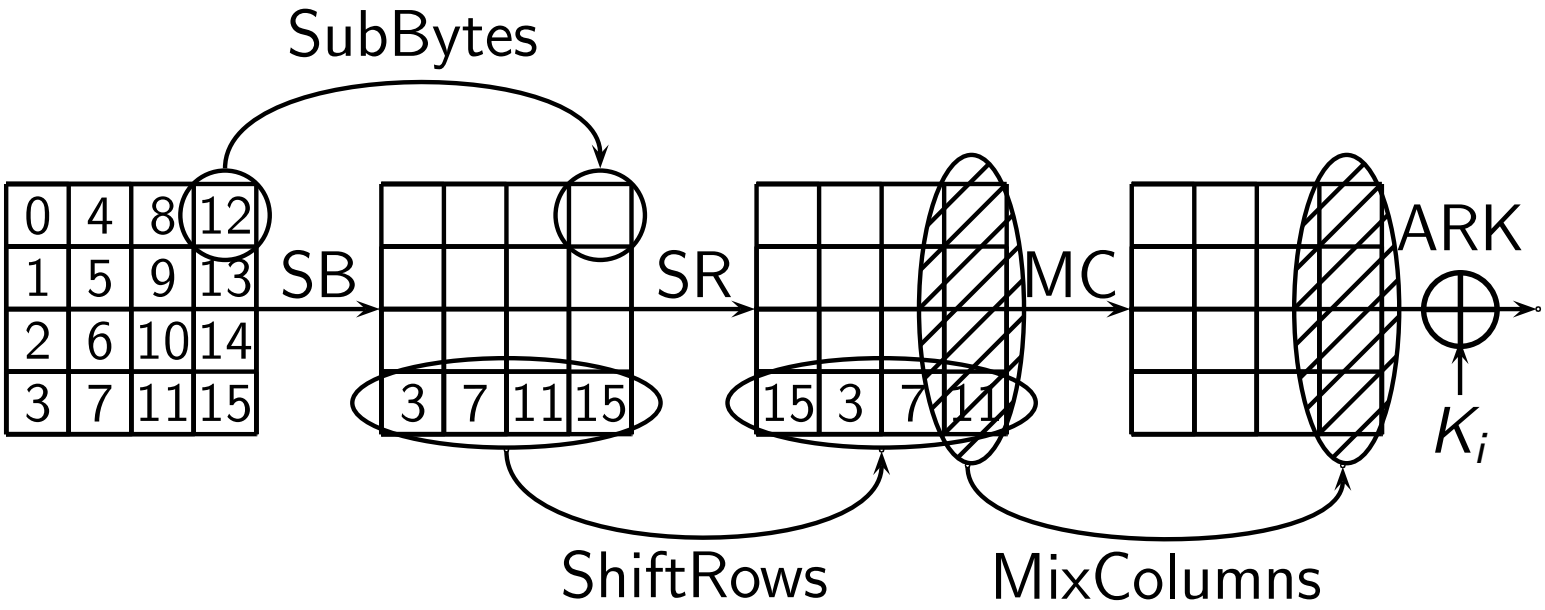
Advanced Encryption Standard

- AES was selected at the end of a similar process to the SHA-3 process by NIST in 2000.
- The selected algorithm, Rijndael, was selected from 15 submissions, of which 5 became known the AES finalists.
- Thoroughly analyzed in many cryptographic settings, and so far withstood all cryptanalytic attempts*.
- Best known attack: 7/10 rounds for 128-bit keys, 8/10 rounds for 192-bit, and 256-bit keys (in the related-key model, the results are 7/9/10 rounds, respectively).

The AES Round Function

- AES is an SPN block cipher.
- The state is treated as a 4x4 byte matrix.
- The round function is composed of four simple operations:
  - SubBytes (SB) — applying the same 8-bit to 8-bit invertible S-box 16 times in parallel on each byte of the state,
  - ShiftRows (SR) — cyclic shift of each row (the i'th row is shifted by i bytes to the left),
  - MixColumns (MC) — multiplication of each column by a constant 4x4 matrix over the field $GF(2^8)$, and
  - AddRoundKey (ARK) — XORing the state with a 128-bit subkey.

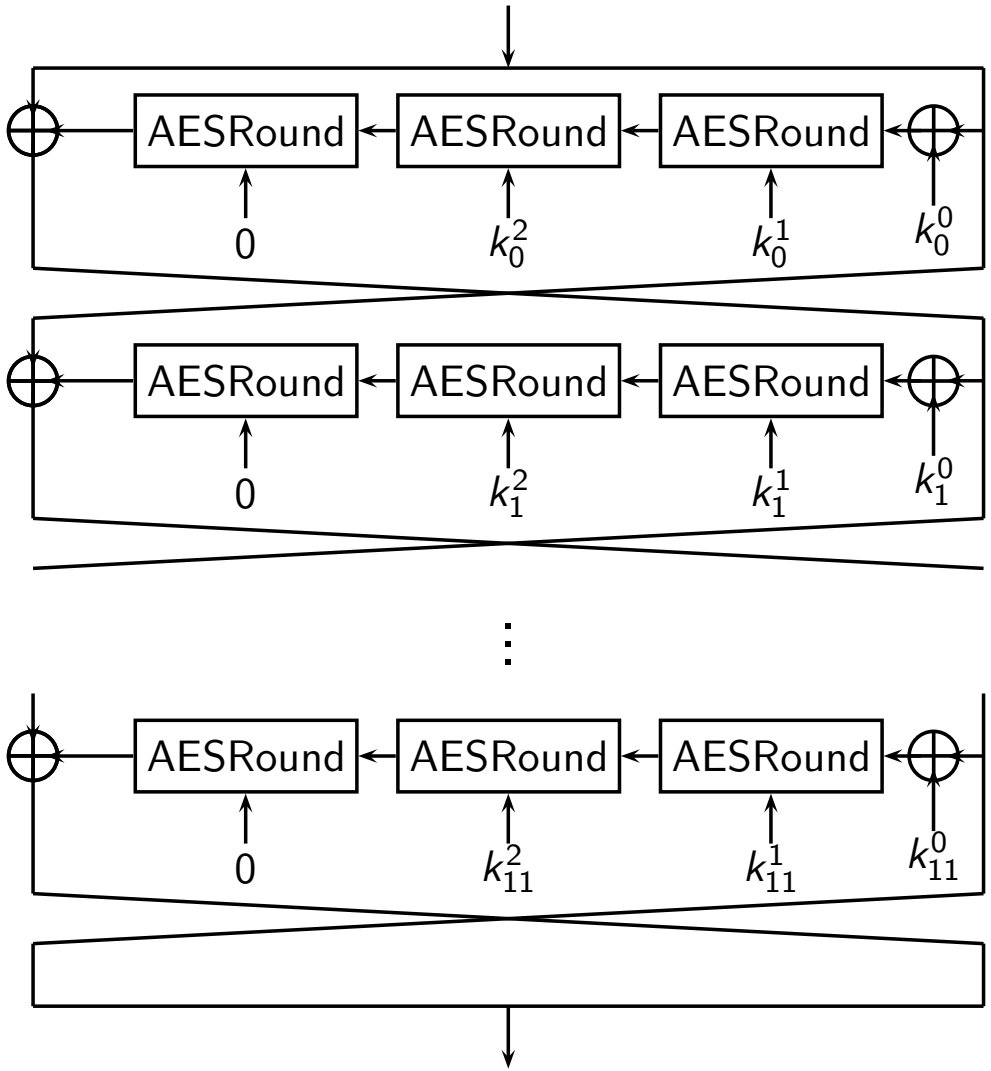
The AES Round Function (cont.)



E^{256} — the Underlying Block Cipher

- Accepts a plaintext (chaining value) of 256 bits.
- Accepts a key (message block, bit counter, and a salt) of 832 bits in total.
- The round function is composed of 3 rounds of AES (with an AddRoundKey operation before the first round, last AddRoundKey operation omitted).
- The message expansion generates 36 128-bit subkeys (12 rounds of E^{256}, each uses 3 round of AES).

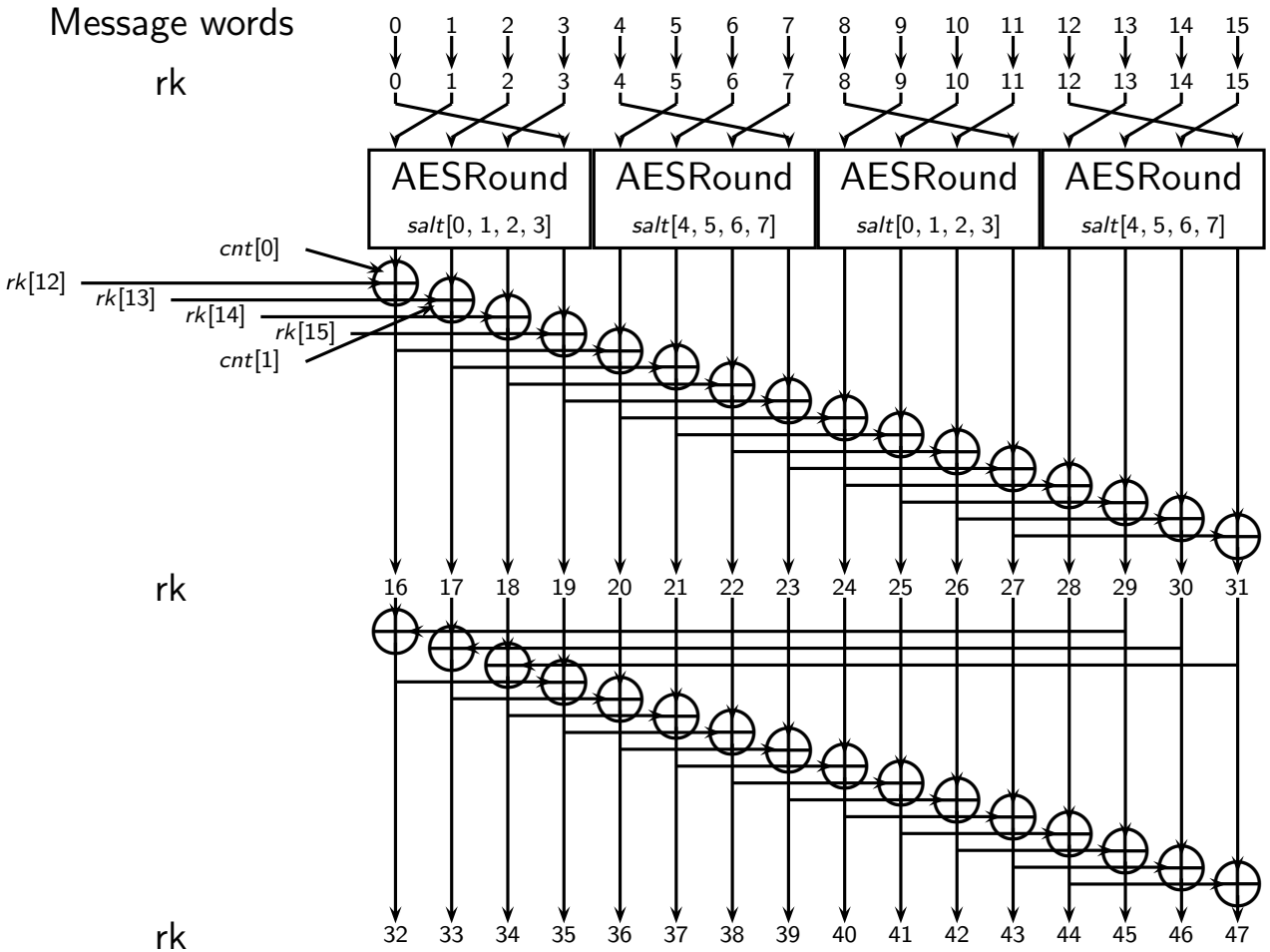
E^{256} — the Underlying Block Cipher (cont.)



The Message Expansion (E^{256} Key Schedule)

- Accepts 832-bit key: 512-bit block, 256-bit salt, 64-bit counter.
- Not all bits are treated equally.
- A combination of an LFSR (for diffusion), and AES rounds (for maximal "confusion" and nonlinearity).

The Message Expansion (E^{256} Key Schedule) (cont.)



The Message Expansion (E^{256} Key Schedule) (cont.)

- The key is expanded into 144 32-bit words.
- The first 16 32-bit words are the message words themselves.
- Then, the following process is repeated four times:
  1. 16 words are generated using applying AES (with the salt as the key) and XORing them with other words.
  2. 16 more words are generated using an LFSR operation on the new 16 words.
- Out of the 144 words, 8 are XORed with the counter words, to ensure that the counter affects the encryption process.

Final Touches — SHAvite-3_{256}

- In order to hash the message M into an m-bit digest, for $m \leq 256$, compute IV_m which is

$$h_0 = IV_m = C_{256}(MIV_{256}, m, 0, 0),$$

- Let $|M|$ be the length of M before padding, measured in bits. Pad the message M according to the padding scheme of HAIFA:
  1. Pad a single bit of 1.
  2. Pad as many 0 bits as needed such that the length of the padded message (with the 1 bit and the 0's) is congruent modulo 512 to 432.
  3. Pad $|M|$ encoded in 64 bits.
  4. Pad m encoded in 16 bits.
- Divide the padded message $pad(M)$ into 512-bit blocks, $pad(M) = M_1||M_2||\ldots||M_l$,

Final Touches — SHAvite-3_{256}

- Set $\#bits \leftarrow 0$.
- Set $h_0 \leftarrow IV_m$.
- For $i = 1, \ldots, \lfloor |M|/512 \rfloor$:
  - Set $\#bits \leftarrow \#bits + 512$.
  - Compute $h_i = C_{256}(h_{i-1}, M_i, \#bits, salt)$.
-
  - If $|M| = 0 \bmod 512$, compute $h_l = C_{256}(h_{l-1}, M_l, 0, salt)$, else
  - If $|M| \bmod 512 \leq 431$, compute $h_l = C_{256}(h_{l-1}, M_l, |M|, salt)$, else
  - Compute $h_{l-1} = C_{256}(h_{l-2}, M_{l-1}, |M|, salt)$, and then compute $h_l = C_{256}(h_{l-1}, M_l, 0, salt)$.
- Output $truncate_m(h_l)$, where $truncate_m(x)$ outputs the m leftmost bits of x, i.e., $x[0]||x[1]|| \ldots$.

SHAvite-3_{512}

- Uses the C_{512} compression function,

SHAvite-3_{512}

- Uses the C_{512} compression function,
- which is a Davies-Meyer transformation of the block cipher E^{512},

SHAvite-3_{512}

- Uses the C_{512} compression function,
- which is a Davies-Meyer transformation of the block cipher E^{512},
- which is a generalized Feistel block cipher, with 14 rounds,

SHAvite-3_{512}

- Uses the C_{512} compression function,
- which is a Davies-Meyer transformation of the block cipher E^{512},
- which is a generalized Feistel block cipher, with 14 rounds,
- where each round is composed of four AES rounds.

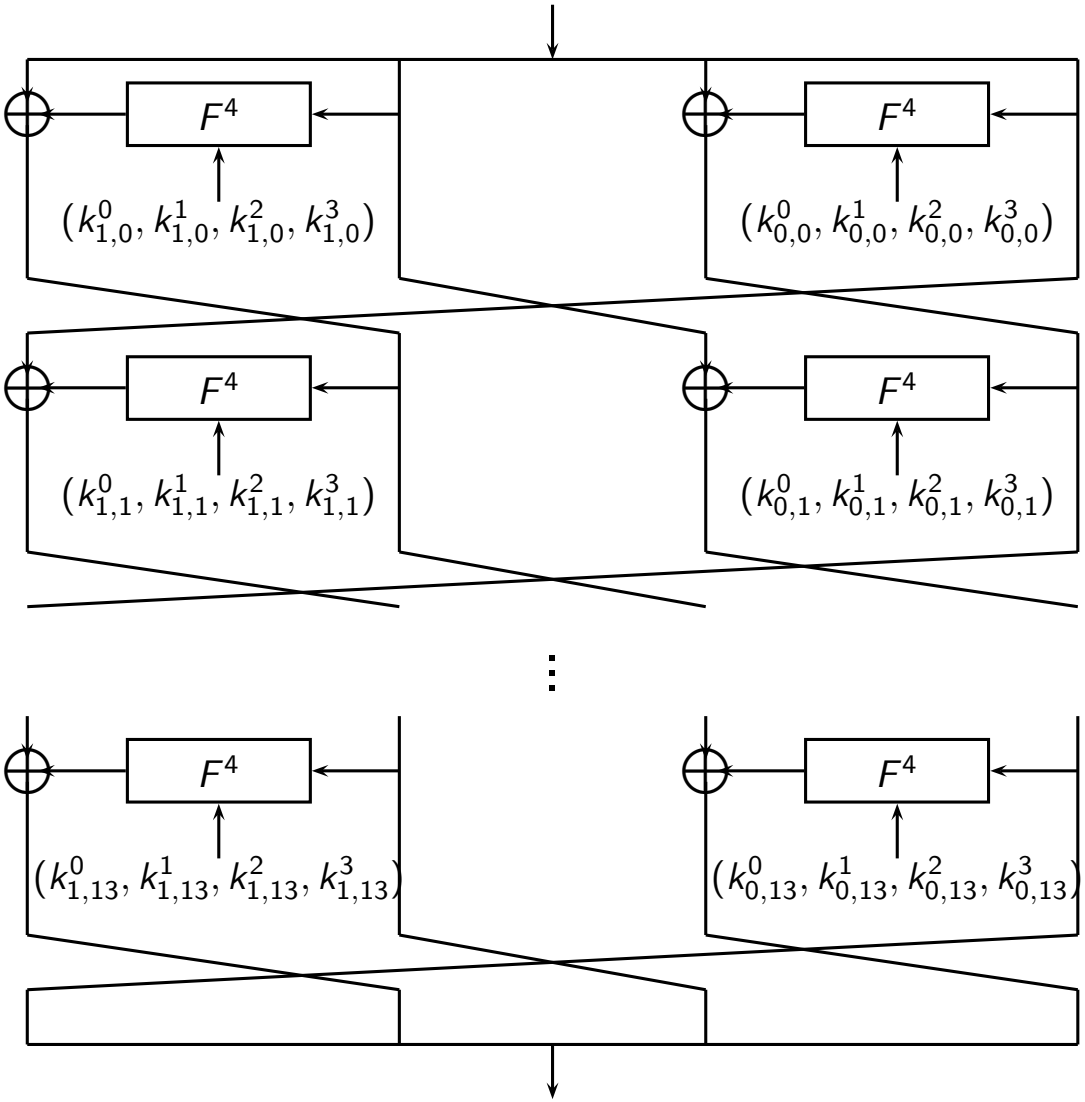
SHAvite-3_{512}

- Uses the C_{512} compression function,
- which is a Davies-Meyer transformation of the block cipher E^{512},
- which is a generalized Feistel block cipher, with 14 rounds,
- where each round is composed of four AES rounds.
- And a completely changed message expansion.

E^{512} — the Underlying Block Cipher

- Accepts a plaintext of 512 bits, and keys of 1664 bits.
- The block cipher is a Generalized Feistel.
- The plaintext is divided into four words of 128 bits each.
- In each of the 14 rounds, two words enter the round function.
- The round function is composed of four AES rounds.

E^{512} — the Underlying Block Cipher (cont.)



How to Pronounce SHAvite-3

SHA-vite SHA-LOSH

What Does it Mean?

- SHA + vite — fast secure hash algorithm.

What Does it Mean?

- SHA + vite — fast secure hash algorithm.
- shavit is a comet in Hebrew

What Does it Mean?

- SHA + vite — fast secure hash algorithm.
- shavit is a comet in Hebrew
- Shavite is a follower of Shiva, god of destruction.

What Does it Mean?

- SHA + vite — fast secure hash algorithm.
- shavit is a comet in Hebrew
- Shavite is a follower of Shiva, god of destruction.
- Shalosh means in 3 in Hebrew.

Outline

Is E^{256} Secure?

- Of course!

Is E^{256} Secure?

- Of course!
- Analyzing three rounds of AES, the maximal expected differential probability is at most 2^{-49}.
- No 2-round iterative characteristics, 3-round iterative characteristics of probability higher than 2^{-98}, ...
- Bound for 9-round characteristic 2^{-294}.
- Similar results hold for linear cryptanalysis/boomerang attacks.
- Impossible differential: 5-round.
- Slide: No can do — counter protects against these.
- Algebraic attacks: full degree after 4 rounds.
- SQUARE: 3-round.

Is E^{512} Secure?

- Of course!

Is E^{512} Secure?

- Of course!
- No 2-round iterative characteristics, 3-round iterative characteristics of probability higher than 2^{-113}, ...
- Bound for 9-round characteristic 2^{-678}.
- Similar results hold for linear cryptanalysis/boomerang attacks.
- Impossible differential: 9-round.
- Slide: No can do — counter protects against these.
- Algebraic attacks: full degree after 4 rounds.
- SQUARE: 3-round.

Extending the Block Cipher Security Results

First Attempt

Computing the maximal expected differential probability of related-key attacks.

Extending the Block Cipher Security Results

First Attempt

Computing the maximal expected differential probability of related-key attacks.

First Attempt Fails

No good methodology for that.

Extending the Block Cipher Security Results

First Attempt

Computing the maximal expected differential probability of related-key attacks.

First Attempt Fails

No good methodology for that.

First Attempt Fails (2)

The attacker controls the keys, he can make sure some differential transitions do happen.

Extending the Security Results — 2nd Attempt

What to do

Consider differentials through the message expansion.

- For a fixed salt, the message expansion can be treated as a block cipher.
- Bound the differential probability of truncated differentials through it.
- Count the number of active S-boxes in the expanded message words, and take into consideration the control the attacker has.
- Each "control" costs either control of the values (message modification) or probability (differential properties).
- Good diffusion in the message expansion — full security for both constructions.

A 3rd Attempt (TBD)

- Collision producing differentials need to go both through the message expansion and the block cipher.
- Each probabilistic event in any of the two should "cost":
  1. Each active byte that enters the actual compression data-path costs at least 8 bits of control.
  2. Each transition of difference column through the MDS matrix in the message expansion — costs control according to the hamming weights.
  3. Each XOR in the message expansion that cancels a difference — costs control.
- Too large of a search space, but gives a very strong upper bound.
- Similar technique may work to "prove" the security of SHA-256 (counting bits rather than active bytes).

The "Theoretical" Security

- HAIFA offers a prefix-free encoding.
- If the compression function is a random oracle the hash function is PRF and PRO (up to the birthday bound).
- If the compression function is a random oracle, the second preimage resistance can be proved to be $O(2^n)$.

Joint work with Charles Bouillaguet, Pierre-Alain Fouque, Sébastien Zimmer.

The Security of AES-Based Submission

How Would an Attack on AES Affects SHAvite-3?

- I don't have a clue.

The Security of AES-Based Submission

How Would an Attack on AES Affects SHAvite-3?

- I don't have a clue.
- You neither.

The Security of AES-Based Submission

How Would an Attack on AES Affects SHAvite-3?

- I don't have a clue.
- You neither.
- It depends on the attack.

The Security of AES-Based Submission (cont.)

Why it is more secure?

- We use a different "key schedule" algorithm.
- We have more rounds on the "critical data path" in the "compression" data-path.
- Attack's target is different.

Why it is less secure?

- The attacker has more control and knowledge on what's going there (open key distinguisher).

Outline

Software Implementation

Hash Function	32-Bit	64-Bit
MD5	7.4	8.8
SHA-1	9.8	9.5
SHA-256	28.8	25.3
SHA-512	77.8	16.9
SHAvite-3_{256} (measured)	35.3	26.7
SHAvite-3_{256} (conjectured)	26.6	18.6
SHAvite-3_{256} (with AES inst.)	< 8	
SHAvite-3_{512} (measured)	55.0	38.2
SHAvite-3_{512} (conjectured)	35.3	28.4
SHAvite-3_{512} (with AES inst.)	< 12	

Expect 1–1.5 cpb improvement if no salts are used.

8-Bit Machines

A Quick and Dirty Approach

- Take best AES implementation.
- Find time required for an AES round.
- Compute number of AES rounds, compute number of other operations.
- And Voilà — an estimate.

8-Bit Machines

A Quick and Dirty Approach

- Take best AES implementation.
- Find time required for an AES round.
- Compute number of AES rounds, compute number of other operations.
- And Voilà — an estimate.

Yes, you can implement SHAvite-3 on an 8-bit machine.

Hardware Implementation

- We looked at four target optimizations for AES: FPGA/ASIC, fastest/smallest.
- Repeating the procedure of the 8-bit machines:

Digest Size	Technology	Size	Throughput
256	ASIC	10.3 Kgates	7.6 Mbps
		55.0 Kgates	604.4 Mbps
	FPGA	510 Slices	1.7 Mbps
		3585 Slices	872.3 Mbps
512	ASIC	18.5 Kgates	4.7 Mbps
		81 Kgates	907.7 Mbps
	FPGA	895 Slices	1.0 Mbps
		7170 Slices	1.12 Gbps

Do note that we used implementation results from 2005. Expect something better. Much much better.

Questions?

Thank you for your attention!